Final Project

Music Rhythm Game

12/9/2022

ECE-4273

Rami Halabi & Trent Famuliner

Description

Our group decided to create a rhythm game for our final project. To meet the required length of the music rhythm game, we used the song *Mary Had a Little Lamb* which has a sequence length of 26 notes. The controller for this game will be a 4x4 keypad to select notes from, while you look at a graphical LCD display that shows the player when to play a certain note. The goal of the game is to successfully play the most amount of notes correctly, on time. Throughout the note sequence, you will hear the song *Mary Had a Little Lamb* depending on the player's accuracy. If you play an incorrect note, a red LED will light up to indicate the player's mistake. The score at the end of the game will be calculated based on the player's accuracy, with the highscore being 2600 points (100 points per note).

Directions for use

To play, press the star symbol (**'*'**) on the keypad. This will start the game. Next, the graphical LCD display will show a note to play. You can identify this note by observing a black dot traverse to the note identifier starting from the left and traveling right. The black dot will hover over a note identifier until it is pressed: '**C**', '**D**', '**E**', or '**G**'. After you press the correct note, A new dot will appear, and then decide which note to press next. This process will keep going until the whole sequence of notes has been played. Upon completion your score will be shown. The highest possible score is 2600. You will be scored on two criteria: Note accuracy and Time accuracy. Try to not play the note early or late for the highest score possible. Also try to play the right note.

Playing the game can be outlined into 4 steps:

1. Press '*' to START.
2. Press keys 'A', 'B', 'C', 'D' to select notes 'C', 'D', 'E', 'G', respectively until the sequence of notes has been played.
3. Observe your score and compete with friends!
4. Should you want to stop the game part way through, the '*' button can also reset the game.

Design Analysis

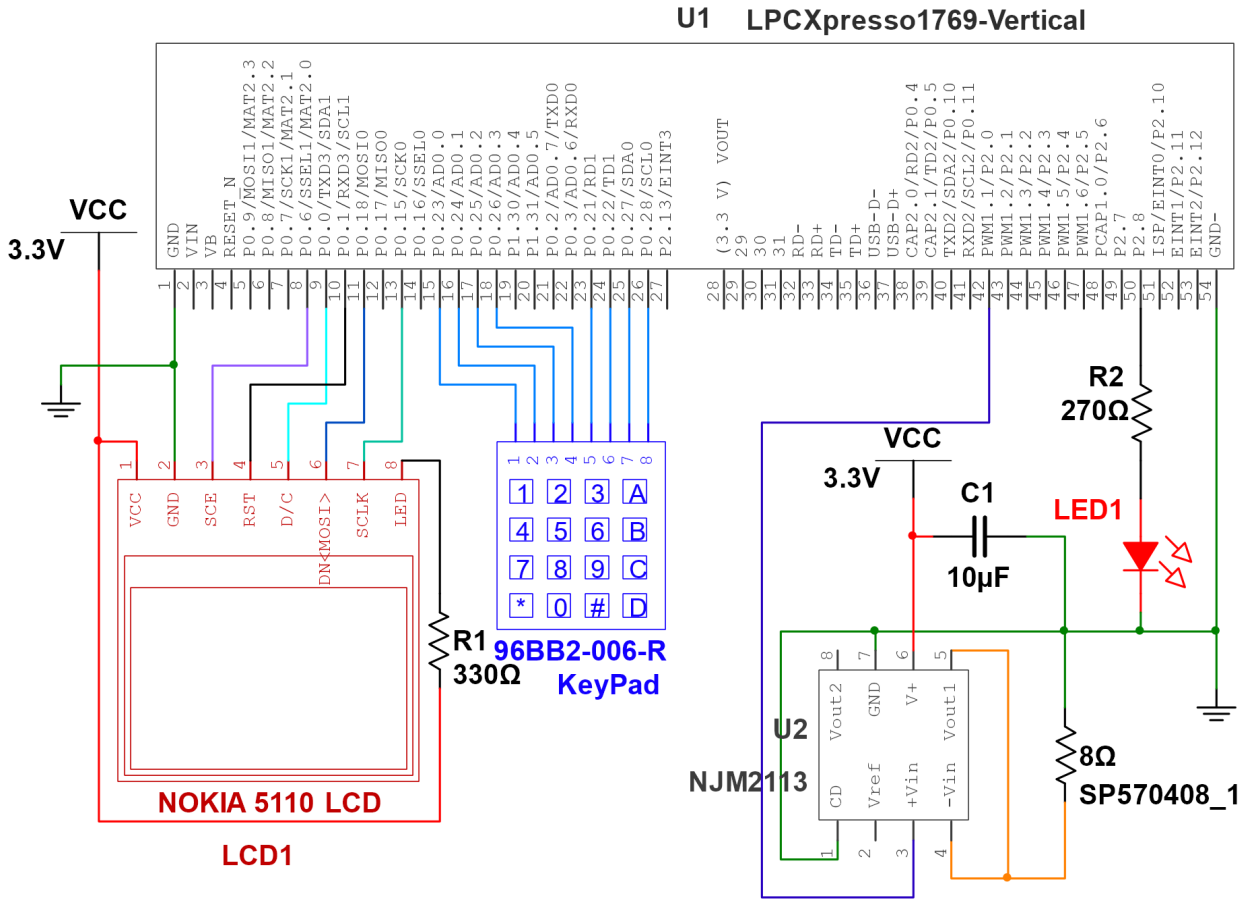
a. **Hardware**

The hardware for this project is fairly simple, and consists of an LCD, a keypad, and an audio amp and Speaker. The only component values needed are a 10μF capacitor across the power rails to keep the amplifier power stable, a 270Ω resistor to limit current to the Red LED, and a 330Ω resistor in series with the LCD LEDs to limit the current to them. The capacitor value was chosen just to be large enough to soak up any voltage fluctuations on the power rail, the 270Ω resistor was chosen to limit the Red LED current to 5 mA, and the LCD resistor value was stated in the documentation for the LCD. The Audio amplifier is just configured in a voltage follower mode as the speaker is plenty loud enough at the 3.3V output of the LPC1769 so there is no need for amplification.

a. **Software**

The Software will use GPIO functions to read the key matrix from the keypad to determine the state of the A, B, C, D, and * keys. The ABCD keys will be used to play the notes, and the * to start and stop the game. Once the game is started, a sequence of notes will be displayed via the LCD which will be communicated with using SPI. The user will then have to press the note keys to match the sequence and then the game will tally up a score based on how far off from the right time they were which will be calculated using timers. This will continue until the song is done and then the score will be tallied and displayed to the LCD. The actual sound waveforms will be implemented using the PWM system to generate the square waves for the notes that the user will play.

Hardware Schematic



Software Code

```
#ifdef __USE_CMSIS
#include  "LPC17xx.h"
#endif
#include <cr_section_macros.h>

//Timer Registers
#define T0TCR (*(volatile unsigned int *)0x40004004)
#define T0TC (*(volatile unsigned int *)0x40004008)
//GPIO Registers
#define FIO0DIR (*(volatile unsigned int *)0x2009c000)
#define FIO0PIN (*(volatile unsigned int *)0x2009c014)
#define FIO2DIR (*(volatile unsigned int *)0x2009C040)
#define FIO2PIN (*(volatile unsigned int *)0x2009C054)
//PINMODE Register
#define PINMODE1 (*(volatile unsigned int *)0x4002C044)
//PINSEL Registers
#define PINSEL0 (*(volatile unsigned int *)0x4002C000)
```

```c
#define PINSEL1 (*(volatile unsigned int *)0x4002C004)
#define PINSEL4 (*(volatile unsigned int *)0x4002C010)
//SPI Registers
#define S0SPCR (*(volatile unsigned int *)0x40020000)
#define S0SPSR (*(volatile unsigned int *)0x40020004)
#define S0SPCCR (*(volatile unsigned int *)0x4002000C)
#define S0SPDR (*(volatile unsigned int *)0x40020008)
// PWM1.1
#define PCLKSEL0 (*(volatile unsigned int *)0x400FC1A8)
#define PWM1MCR (*(volatile unsigned int *)0x40018014)
#define PWM1TCR (*(volatile unsigned int *)0x40018004)
#define PWM1PCR (*(volatile unsigned int *)0x4001804C)
#define PWM1MR0 (*(volatile unsigned int *)0x40018018)
#define PWM1MR1 (*(volatile unsigned int *)0x4001801C)
#define PWM1LER (*(volatile unsigned int *)0x40018050)
#define PWM1PR (*(volatile unsigned int *)0x4001800C)
//busywork variable to waste time
volatile int busywork;

//Default Screen Constant
const unsigned char DefaultScreen [] = {
0x00, 0xF0, 0x08, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x84, 0x02, 0x02, 0x02, 0x02,
0x84, 0x78, 0x00, 0x00, 0x08, 0x88, 0x89, 0x89, 0x09, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08,
0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08,
0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08,
0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08,
0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x88, 0x48,
0x29, 0x29, 0x29, 0x29, 0x48, 0x88, 0x08, 0x08, 0x80, 0x9F, 0x90, 0x90, 0x8F, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x87, 0x88, 0x90, 0x90, 0x90, 0x90, 0x88, 0x87, 0x80, 0x80, 0x00, 0xF8, 0x48, 0x48,
0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x84, 0x02, 0x02, 0x02, 0x02, 0x84, 0x78, 0x00, 0x00,
0x08, 0x09, 0x89, 0x89, 0x89, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08,
0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08,
0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08,
0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08,
0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x88, 0x48, 0x29, 0x29, 0x29, 0x29,
0x48, 0x88, 0x08, 0x08, 0x80, 0x8F, 0x90, 0x94, 0x9C, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x87, 0x88,
0x90, 0x90, 0x90, 0x90, 0x88, 0x87, 0x80, 0x80,
};
const unsigned char ScoreScreen [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xCF, 0xDF, 0x99, 0x99, 0x99, 0x99,
0x99, 0x99, 0x99, 0xFB, 0xFB, 0x00, 0x00, 0x00, 0x00, 0xFE, 0xFE, 0x02, 0x02, 0x02, 0x02, 0x02,
0xCE, 0xCE, 0x00, 0x00, 0x00, 0xFE, 0xFE, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0xFE, 0xFE, 0x00,
0x00, 0x00, 0x00, 0xFE, 0xFE, 0x22, 0x22, 0x62, 0xE2, 0xE2, 0xBE, 0x3E, 0x00, 0x00, 0x00, 0xFE,
```

```
0xFE, 0x22, 0x22, 0x22, 0x22, 0x02, 0x02, 0x02, 0x00, 0x00, 0x00, 0xCE, 0xCE, 0xCE, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01,
0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01,
0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01,
0x01, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x01,
0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x02, 0x02,
0x02, 0x02, 0x00, 0x00, 0x02, 0x02, 0x02, 0x02, 0x02, 0x00, 0x00, 0x02, 0x02, 0x02, 0x02, 0x02,
0x00, 0x00, 0x02, 0x02, 0x02, 0x02, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
const unsigned char Nums[][5] =
{{0x3e, 0x51, 0x49, 0x45, 0x3e} // 0
,{0x00, 0x42, 0x7f, 0x40, 0x00} // 1
,{0x42, 0x61, 0x51, 0x49, 0x46} // 2
,{0x21, 0x41, 0x45, 0x4b, 0x31} // 3
,{0x18, 0x14, 0x12, 0x7f, 0x10} // 4
,{0x27, 0x45, 0x45, 0x45, 0x39} // 5
,{0x3c, 0x4a, 0x49, 0x49, 0x30} // 6
,{0x01, 0x71, 0x09, 0x05, 0x03} // 7
,{0x36, 0x49, 0x49, 0x49, 0x36} // 8
,{0x06, 0x49, 0x49, 0x29, 0x1e} // 9
};

const unsigned char blockCE = {0x78};
const unsigned char blankCE = {0x00};
const unsigned char circblockCE = {0x7A};
const unsigned char blockDG [] = {0x88,0x87};
const unsigned char blankDG [] = {0x08,0x80};
const unsigned char circblockDG [] = {0xA9,0x97};

int abs(int num)
{
	if(num>= 0)
	{
		return num;
	}
	else
	{
		return -num;
	}
}
void wait_us(float us)
{
    int ticks = 0.367776*us - 3.113;
    for (int count=0; count<ticks; count++){} // do nothing
}
```

```
void wait(float sec)
{
    //convert sec to us
    int us = sec*1000000;
    wait_us(us);
}
void wait_ms(int ms)
{
    //convert ms to us
    int us = ms*1000;
    wait_us(us);
}

void SetRedLed(int state)
{
    if(state)
        FIO2PIN |= (1<<8);
    else
        FIO2PIN &= ~(1<<8);
    busywork++;
}
void SetDC(int state)
{
    if(state)
        FIO0PIN |= (1<<0);
    else
        FIO0PIN &= ~(1<<0);
    busywork++;
}
void SetRes(int state)
{
    if(state)
        FIO0PIN |= (1<<1);
    else
        FIO0PIN &= ~(1<<1);
    busywork++;
}
void SetSel(int state)
{
    if(state)
        FIO0PIN |= (1<<6);
    else
        FIO0PIN &= ~(1<<6);
    busywork++;
}
void LcdWrite(int dc, int data)
{
    SetDC(dc);
    SetSel(0);
    S0SPDR |= data;
    while(!((S0SPSR>>7)&1))
    {
        //Wait for SPIF to return to 1
    }
    SetSel(1);
```

```
}
void LcdDefault(void)
{
    //reset XY
    LcdWrite(0, 0x80);
    LcdWrite(0, 0x40);
    for(int idx = 0; idx <504; idx++)
    {
        LcdWrite(1, DefaultScreen[idx]);
    }
}
void LcdBlank(void)
{
    //reset XY
    //LcdWrite(0, 0x80);
    //LcdWrite(0, 0x40);
    for(int idx = 0; idx <504; idx++)
    {
        LcdWrite(1, 0x00);
    }
}
void LcdScoreScreen(int score)
{
       int thou, hund, ten, one;
       thou = score / 1000;
       hund = (score%1000) / 100;
       ten = ((score%1000)%100) / 10;
       one = (((score%1000)%100)%10);
       //reset XY
       LcdWrite(0, 0x80);
       LcdWrite(0, 0x40);
       for(int idx = 0; idx <504; idx++)
       {
              LcdWrite(1, ScoreScreen[idx]);
       }
       //set XY
       LcdWrite(0, 0x9D); //x = 29
       LcdWrite(0, 0x43); //y = 3
       for(int idx = 0; idx <5; idx++)
       {
              LcdWrite(1, Nums[thou][idx]);
       }
       LcdWrite(1, 0x00);
       LcdWrite(1, 0x00);
       for(int idx = 0; idx <5; idx++)
       {
              LcdWrite(1, Nums[hund][idx]);
       }
       LcdWrite(1, 0x00);
       LcdWrite(1, 0x00);
       for(int idx = 0; idx <5; idx++)
       {
              LcdWrite(1, Nums[ten][idx]);
       }
       LcdWrite(1, 0x00);
```

```
        LcdWrite(1, 0x00);
        for(int idx = 0; idx <5; idx++)
        {
                LcdWrite(1, Nums[one][idx]);
        }
}
// check if button 'A' is pushed.
int checkG(void){

    // configure terminal pin 5 as output and make it go high
    FIO0DIR |= (1<<21); // Terminal pin 5 to output
    FIO0PIN |= (1<<21); // Make terminal pin 5 output high
    PINMODE1 |= (0<<11) | (0<<10); // Terminal pin 5 PullUp
    wait_ms(1);

    // store result
    int result = (FIO0PIN >> 23) & 1; // read pin 1

    // set terminal pin 5 back to input
    FIO0DIR &= ~(1<<21);
    PINMODE1 |= (1<<11) | (1<<10); // Terminal pin 5 PullDown
    wait_ms(1);

    // check result
    return result;
}

// check if button 'B' is pushed.
int checkE(void){

    // configure terminal pin 6 as output and make it go high
    FIO0DIR |= (1<<22); // Terminal pin 6 to output
    FIO0PIN |= (1<<22); // Make terminal pin 6 output high
    PINMODE1 |= (0<<13) | (0<<12); // Terminal pin 6 PullUp
    wait_ms(1);

    // store result
    int result = (FIO0PIN >> 23) & 1; // read pin 1

    // set terminal pin 6 back to input
    FIO0DIR &= ~(1<<22);
    PINMODE1 |= (1<<13) | (1<<12); // Terminal pin 6 PullDown
    wait_ms(1);

    // check result
    return result;
}

// check if button 'C' is pushed.
int checkD(void){


    // configure terminal pin 3 as output and make it go high
    FIO0DIR |= (1<<23); // Terminal pin 3 to output
    FIO0PIN |= (1<<23); // Make terminal pin 3 output high
```

```
    PINMODE1 |= (0<<15) | (0<<14); // Terminal pin 3 PullUp
    wait_ms(1);

    // store result
    int result = (FIO0PIN >> 27) & 1; // read pin 7

    // set terminal pin 3 back to input
    FIO0DIR &= ~(1<<23);
    PINMODE1 |= (0<<15) | (0<<14); // Terminal pin 3 PullDown
    wait_ms(1);

    // check result
    return result;
}

// check if button 'D' is pushed.
int checkC(void){

    // configure terminal pin 4 as output and make it go high
    FIO0DIR |= (1<<23); // Terminal pin 4 to output
    FIO0PIN |= (1<<23); // Make terminal pin 4 output high
    PINMODE1 |= (0<<15) | (0<<14); // Terminal pin 4 PullUp
    wait_ms(1);

    // store result
    int result = (FIO0PIN >> 28) & 1;

    // set terminal pin 4 back to input
    FIO0DIR &= ~(1<<23);
    PINMODE1 |= (1<<15) | (1<<14); // Terminal pin 4 PullDown
    wait_ms(1);

    // check result
    return result;
}

// check if button '*' is pushed.
int checkStar(void){

    // configure terminal pin 5 as output and make it go high
    FIO0DIR |= (1<<21); // Terminal pin 5 to output
    FIO0PIN |= (1<<21); // Make terminal pin 5 output high
    PINMODE1 |= (0<<11) | (0<<10); // Terminal pin 5 PullUp
    wait_ms(1);

    // store result
    int result = (FIO0PIN >> 26) & 1; // read pin 4

    // set terminal pin 5 back to input
    FIO0DIR &= ~(1<<21);
    PINMODE1 |= (1<<11) | (1<<10); // Terminal pin 5 PullDown
    wait_ms(1);

    // check result
    return result;
```

```
}

// user manual pg 520
void init_PWM(int MR0, int ms)
{
    // Initialize PWM to 500KHz (4MHZ/8)
    PWM1PR = 0;
    PCLKSEL0 |= (1<<12); // PWM PCLK = CCLK/8
    PCLKSEL0 |= (1<<13);
    PWM1MR0 = MR0; // PCLK cycle period
    PINSEL4 |= (1<<0); // Configure P2.0 as PWM1.1
    PWM1MCR = (1<<1); // Reset counter on MR0 match
    PWM1PCR = (1<<9); // Single edge mode and enable PWM1.1 only
    PWM1TCR = (1<<0) | (1<<3); // Enable counter and PWM mode
    PWM1MR1 = PWM1MR0/2; // change MR1 to output new 8-bit D sample value
    PWM1LER = (1<<1) | (1<<0); // set bit 1 to use new MR1 value
    wait_ms(1); // avoid changing the D too fast
    wait_ms(ms); //play note for ms milliseconds
    PWM1MR1 = 0; // change MR1 to output new 8-bit D sample value
    PWM1LER = (1<<1) | (1<<0); // set bit 1 to use new MR1 value
}
void play_note(char note, int ms){

    int MR0 = 0;
    // Determine PWM1MR0 value in PWM from determined note frequency
    switch (note)
    {
    case 'C':
        MR0 = 1929; // 'C'-tone frequency: 261.63Hz
        break;
    case 'D':
        MR0 = 1720; // 'D'-tone frequency 293.66Hz
        break;
    case 'E':
        MR0 = 1532; // 'E'-tone frequency 329.63Hz
        break;
    case 'G':
        MR0 = 1288; // 'G'-tone frequency 392Hz
        break;
    default:
        break;
    }

    // Play note
    init_PWM(MR0, ms);
}
void init_keyPad(void){

        // declare key pad inputs

        // columns
        FIO0DIR &= ~(1<<23); // Terminal pin 1
        FIO0DIR &= ~(1<<24); // Terminal pin 2
        FIO0DIR &= ~(1<<25); // Terminal pin 3
        FIO0DIR &= ~(1<<26); // Terminal pin 4
```

```
        // rows
        FIO0DIR &= ~(1<<21); // Terminal pin 5
        FIO0DIR &= ~(1<<22); // Terminal pin 6
        FIO0DIR &= ~(1<<27); // Terminal pin 7
        FIO0DIR &= ~(1<<28); // Terminal pin 8

        // assign pull-down to inputs
        PINMODE1 |= (1<<15) | (1<<14); // Terminal pin 1 PullDown
        PINMODE1 |= (1<<17) | (1<<16); // Terminal pin 2 PullDown
        PINMODE1 |= (1<<19) | (1<<18); // Terminal pin 3 PullDown
        PINMODE1 |= (1<<21) | (1<<20); // Terminal pin 4 PullDown
        PINMODE1 |= (1<<11) | (1<<10); // Terminal pin 5 PullDown
        PINMODE1 |= (1<<13) | (1<<12); // Terminal pin 6 PullDown
        PINMODE1 |= (1<<23) | (1<<22); // Terminal pin 7 PullDown
        PINMODE1 |= (1<<25) | (1<<24); // Terminal pin 8 PullDown

        wait_ms(5);
}

// initialize GPIO, SPI, and Setup LCD
void initializePorts(void)
{
    init_keyPad();
    FIO2DIR |= (1<<8);  // REDLED to output
    FIO0DIR |= (1<<0);  //D/C
    FIO0DIR |= (1<<1);  //Reset
    FIO0DIR |= (1<<6);  //Sel

    PINSEL0 |= (1<<31) | (1<<30); //Set pin 12 to SCK
    PINSEL1 |= (1<<5) | (1<<4); //Set MOSI

    S0SPCR |= (1<<5); //Configure SPI to master
    S0SPCCR |= 0b1010; //Set clk to 100kHz
    busywork++;
    SetRes(0);
    wait_us(1);
    SetRes(1);
    LcdWrite(0, 0x21 );  // LCD Extended Commands.
    LcdWrite(0, 0xB1 );  // Set LCD Vop (Contrast).
    LcdWrite(0, 0x04 );  // Set Temp coefficent. //0x04
    LcdWrite(0, 0x14 );  // LCD bias mode 1:48. //0x13
    LcdWrite(0, 0x20 );  // LCD Basic Commands
    LcdWrite(0, 0x0C );  // LCD in normal mode.
}
void LcdBlockC(int num)
{
    int xcord = 16 + 20*num;
    int xdat = (0x80) | (xcord);
    LcdWrite(0, xdat); //set x coord to xcord
    LcdWrite(0, 0x40); //set y coord to 0
    for(int idx = 0; idx <4; idx++)
    {
        if(num == 3)
        {
```

```
            LcdWrite(1, circblockCE);
        }
        else
        {
            LcdWrite(1, blockCE);
        }
    }
    if(num != 0)
    {
        xdat = (0x80) | (xcord-20);
        LcdWrite(0, xdat); //set x coord to xcord
        LcdWrite(0, 0x40); //set y coord to 0
        for(int idx = 0; idx <4; idx++)
        {
            LcdWrite(1, blankCE);
        }
    }
}
void LcdBlockE(int num)
{
    int xcord = 16 + 20*num;
    int xdat = (0x80) | (xcord);
    LcdWrite(0, xdat); //set x coord to xcord
    LcdWrite(0, 0x43); //set y coord to 3
    for(int idx = 0; idx <4; idx++)
    {
        if(num == 3)
        {
            LcdWrite(1, circblockCE);
        }
        else
        {
            LcdWrite(1, blockCE);
        }
    }
    if(num != 0)
    {
        xdat = (0x80) | (xcord-20);
        LcdWrite(0, xdat); //set x coord to xcord
        LcdWrite(0, 0x43); //set y coord to 3
        for(int idx = 0; idx <4; idx++)
        {
            LcdWrite(1, blankCE);
        }
    }
}
void LcdBlockD(int num)
{
    int xcord = 16 + 20*num;
    int xdat = (0x80) | (xcord);
    LcdWrite(0, xdat); //set x coord to xcord
    LcdWrite(0, 0x41); //set y coord to 1
    for(int idx = 0; idx <4; idx++)
    {
        if(num == 3)
```

```
        {
            LcdWrite(1, circblockDG[0]);
        }
        else
        {
            LcdWrite(1, blockDG[0]);
        }
    }
    LcdWrite(0, xdat); //set x coord to xcord
    LcdWrite(0, 0x42); //set y coord to 2
    for(int idx = 0; idx <4; idx++)
    {
        if(num == 3)
        {
            LcdWrite(1, circblockDG[1]);
        }
        else
        {
            LcdWrite(1, blockDG[1]);
        }
    }
    if(num != 0)
    {
        xdat = (0x80) | (xcord-20);
        LcdWrite(0, xdat); //set x coord to xcord
        LcdWrite(0, 0x41); //set y coord to 1
        for(int idx = 0; idx <4; idx++)
        {
            LcdWrite(1, blankDG[0]);
        }
        LcdWrite(0, xdat); //set x coord to xcord
        LcdWrite(0, 0x42); //set y coord to 2
        for(int idx = 0; idx <4; idx++)
        {
            LcdWrite(1, blankDG[1]);
        }
    }
}
void LcdBlockG(int num)
{
    int xcord = 16 + 20*num;
    int xdat = (0x80) | (xcord);
    LcdWrite(0, xdat); //set x coord to xcord
    LcdWrite(0, 0x44); //set y coord to 4
    for(int idx = 0; idx <4; idx++)
    {
        if(num == 3)
        {
            LcdWrite(1, circblockDG[0]);
        }
        else
        {
            LcdWrite(1, blockDG[0]);
        }
    }
```

```
    LcdWrite(0, xdat); //set x coord to xcord
    LcdWrite(0, 0x45); //set y coord to 5
    for(int idx = 0; idx <4; idx++)
    {
        if(num == 3)
        {
            LcdWrite(1, circblockDG[1]);
        }
        else
        {
            LcdWrite(1, blockDG[1]);
        }
    }
    if(num != 0)
    {
        xdat = (0x80) | (xcord-20);
        LcdWrite(0, xdat); //set x coord to xcord
        LcdWrite(0, 0x44); //set y coord to 4
        for(int idx = 0; idx <4; idx++)
        {
            LcdWrite(1, blankDG[0]);
        }
        LcdWrite(0, xdat); //set x coord to xcord
        LcdWrite(0, 0x45); //set y coord to 5
        for(int idx = 0; idx <4; idx++)
        {
            LcdWrite(1, blankDG[1]);
        }
    }
}
int Csequence(void)
{
    int score = 0;
    LcdDefault();
    int blocknum = 0;
    T0TCR &= ~(1<<0);   //Make Sure timer is stopped
    T0TCR |= (1<<1);    //reset Timer
    while(T0TC != 0){}  //wait till timer is 0
    T0TCR &= ~(1<<1);   //clear reset
    T0TCR |= (1<<0);    //Start Timer

    //move blocks across screen
    LcdBlockC(blocknum);
    blocknum++;
    while(!checkStar())
    {
        if((T0TC>125000)&&(blocknum == 1))
        {
            LcdBlockC(blocknum);
            blocknum++;
        }
        if((T0TC>250000)&&(blocknum == 2))
        {
            LcdBlockC(blocknum);
            blocknum++;
```

```
        }
        if((T0TC>375000)&&(blocknum == 3))
        {
            LcdBlockC(blocknum);
            blocknum++;
        }
        if(checkC())
        {
            if((400000< T0TC)&&(T0TC<600000))
                score = 100;
            else
                score = 100 - abs(T0TC-500000)/10000;
            play_note('C', 250);
            if(score <0)
                score = 0;
            return score;
        }
        if(checkD())
        {
            SetRedLed(1);
            play_note('D', 250);
            SetRedLed(0);
            return 0;
        }
        if(checkE())
        {
            SetRedLed(1);
            play_note('E', 250);
            SetRedLed(0);
            return 0;
        }
        if(checkG())
        {
            SetRedLed(1);
            play_note('G', 250);
            SetRedLed(0);
            return 0;
        }
    }
    return -2600;
}
int Dsequence(void)
{
    int score = 0;
    LcdDefault();
    int blocknum = 0;
    T0TCR &= ~(1<<0);   //Make Sure timer is stopped
    T0TCR |= (1<<1);    //reset Timer
    while(T0TC != 0){}  //wait till timer is 0
    T0TCR &= ~(1<<1);   //clear reset
    T0TCR |= (1<<0);    //Start Timer

    //move blocks across screen
    LcdBlockD(blocknum);
    blocknum++;
```

```
    while(!checkStar())
    {
        if((T0TC>125000)&&(blocknum == 1))
        {
            LcdBlockD(blocknum);
            blocknum++;
        }
        if((T0TC>250000)&&(blocknum == 2))
        {
            LcdBlockD(blocknum);
            blocknum++;
        }
        if((T0TC>375000)&&(blocknum == 3))
        {
            LcdBlockD(blocknum);
            blocknum++;
        }
        if(checkC())
        {
            SetRedLed(1);
            play_note('C', 250);
            SetRedLed(0);
            return 0;
        }
        if(checkD())
        {
            if((400000< T0TC)&&(T0TC<600000))
                score = 100;
            else
                score = 100 - abs(T0TC-500000)/10000;
            play_note('D', 250);
            if(score <0)
                score = 0;
            return score;
        }
        if(checkE())
        {
            SetRedLed(1);
            play_note('E', 250);
            SetRedLed(0);
            return 0;
        }
        if(checkG())
        {
            SetRedLed(1);
            play_note('G', 250);
            SetRedLed(0);
            return 0;
        }
    }
    return -2600;
}
int Esequence(void)
{
    int score = 0;
```

```
LcdDefault();
int blocknum = 0;
T0TCR &= ~(1<<0);   //Make Sure timer is stopped
T0TCR |= (1<<1);    //reset Timer
while(T0TC != 0){}  //wait till timer is 0
T0TCR &= ~(1<<1);   //clear reset
T0TCR |= (1<<0);    //Start Timer

//move blocks across screen
LcdBlockE(blocknum);
blocknum++;
while(!checkStar())
{
    if((T0TC>125000)&&(blocknum == 1))
    {
        LcdBlockE(blocknum);
        blocknum++;
    }
    if((T0TC>250000)&&(blocknum == 2))
    {
        LcdBlockE(blocknum);
        blocknum++;
    }
    if((T0TC>375000)&&(blocknum == 3))
    {
        LcdBlockE(blocknum);
        blocknum++;
    }
    if(checkC())
    {
        SetRedLed(1);
        play_note('C', 250);
        SetRedLed(0);
        return 0;
    }
    if(checkD())
    {
        SetRedLed(1);
        play_note('D', 250);
        SetRedLed(0);
        return 0;
    }
    if(checkE())
    {
        if((400000< T0TC)&&(T0TC<600000))
            score = 100;
        else
            score = 100 - abs(T0TC-500000)/10000;
        play_note('E', 250);
        if(score <0)
            score = 0;
        return score;
    }
    if(checkG())
    {
```

```
            SetRedLed(1);
            play_note('G', 250);
            SetRedLed(0);
            return 0;
        }
    }
    return -2600;

}
int Gsequence(void)
{
    int score = 0;
    LcdDefault();
    int blocknum = 0;
    T0TCR &= ~(1<<0);   //Make Sure timer is stopped
    T0TCR |= (1<<1);    //reset Timer
    while(T0TC != 0){}  //wait till timer is 0
    T0TCR &= ~(1<<1);   //clear reset
    T0TCR |= (1<<0);    //Start Timer

    //move blocks across screen
    LcdBlockG(blocknum);
    blocknum++;
    while(!checkStar())
    {
        if((T0TC>125000)&&(blocknum == 1))
        {
            LcdBlockG(blocknum);
            blocknum++;
        }
        if((T0TC>250000)&&(blocknum == 2))
        {
            LcdBlockG(blocknum);
            blocknum++;
        }
        if((T0TC>375000)&&(blocknum == 3))
        {
            LcdBlockG(blocknum);
            blocknum++;
        }
        if(checkC())
        {
            SetRedLed(1);
            play_note('C', 250);
            SetRedLed(0);
            return 0;
        }
        if(checkD())
        {
            SetRedLed(1);
            play_note('G', 250);
            SetRedLed(0);
            return 0;
        }
        if(checkE())
```

```
        {
            SetRedLed(1);
            play_note('E', 250);
            SetRedLed(0);
            return 0;
        }
        if(checkG())
        {
            if((400000< T0TC)&&(T0TC<600000))
                score = 100;
            else
                score = 100 - abs(T0TC-500000)/10000;
            play_note('G', 250);
            if(score <0)
                score = 0;
            return score;
        }
    }
    return -2600;

}
int playGame(void)
{
    int score = 0;
    score += Esequence();   //E
    if(score<0){return 0;}
    score += Dsequence();   //D
    if(score<0){return 0;}
    score += Csequence();   //C
    if(score<0){return 0;}
    score += Dsequence();   //D
    if(score<0){return 0;}
    score += Esequence();   //E
    if(score<0){return 0;}
    score += Esequence();   //E
    if(score<0){return 0;}
    score += Esequence();   //E
    if(score<0){return 0;}
    wait_ms(250);

    score += Dsequence();   //D
    if(score<0){return 0;}
    score += Dsequence();   //D
    if(score<0){return 0;}
    score += Dsequence();   //D
    if(score<0){return 0;}
    wait_ms(250);
    score += Esequence();   //E
    if(score<0){return 0;}
    score += Gsequence();   //G
    if(score<0){return 0;}
    score += Gsequence();   //G
    if(score<0){return 0;}
    wait_ms(250);
```

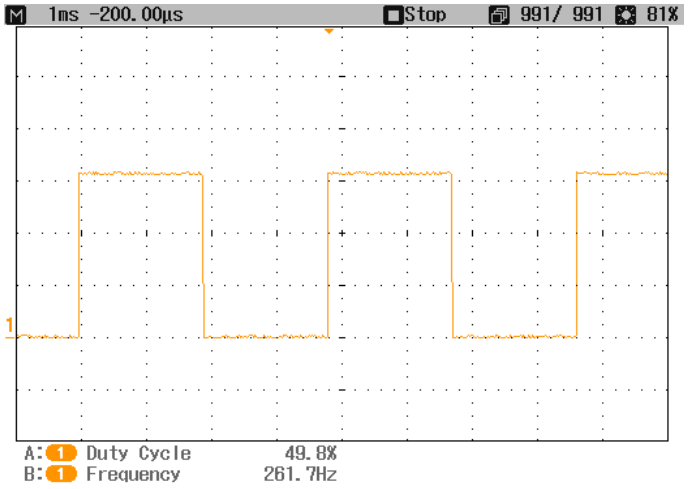
```
    score += Esequence();   //E
    if(score<0){return 0;}
    score += Dsequence();   //D
    if(score<0){return 0;}
    score += Csequence();   //C
    if(score<0){return 0;}
    score += Dsequence();   //D
    if(score<0){return 0;}
    score += Esequence();   //E
    if(score<0){return 0;}
    score += Esequence();   //E
    if(score<0){return 0;}
    score += Esequence();   //E
    if(score<0){return 0;}
    score += Esequence();   //E
    if(score<0){return 0;}

    score += Dsequence();   //D
    if(score<0){return 0;}
    score += Dsequence();   //D
    if(score<0){return 0;}
    score += Esequence();   //E
    if(score<0){return 0;}
    score += Dsequence();   //D
    if(score<0){return 0;}
    score += Csequence();   //C
    if(score<0){return 0;}
    return score;
}
int main(void)
{
    int lastscore = 0;
    initializePorts();
    LcdDefault();
    while(1)
    {
        if(checkStar())
        {
             while(checkStar()){}
             wait_ms(25);
            lastscore = playGame();
            if(checkStar)
            {
             while(checkStar()){}
             wait_ms(25);
            }
            LcdScoreScreen(lastscore);
        }
    }
    return 0;
}
```

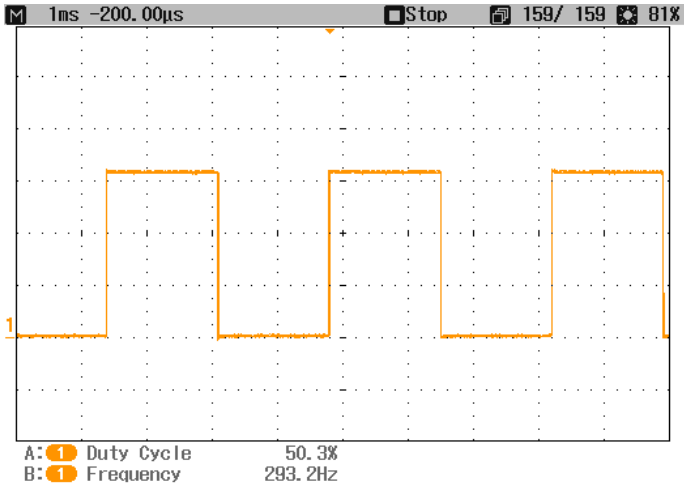
Supporting Documentation

Oscilloscope Screenshots:

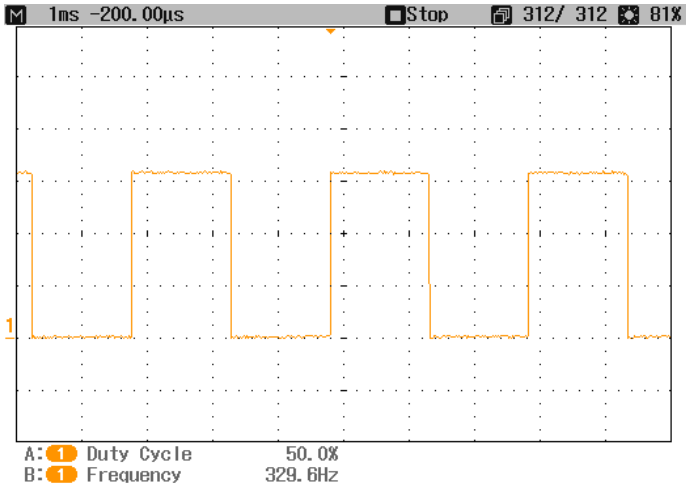
‘C’ - tone Frequency: 261.63 Hz



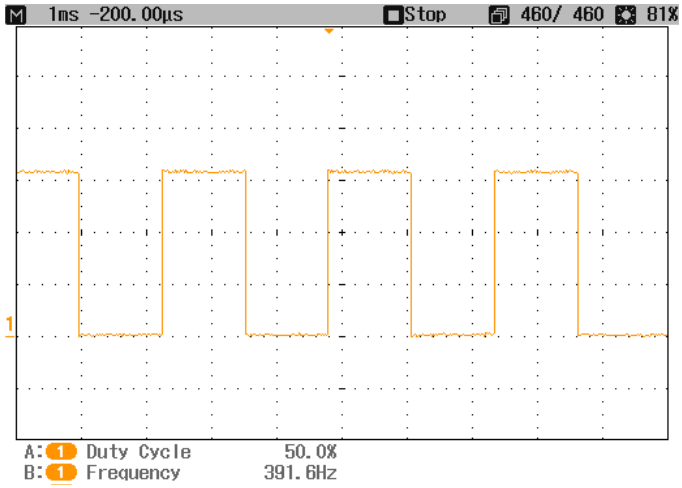
‘D’ - tone Frequency: 293.66 Hz



‘E’ - tone Frequency: 329.63 Hz

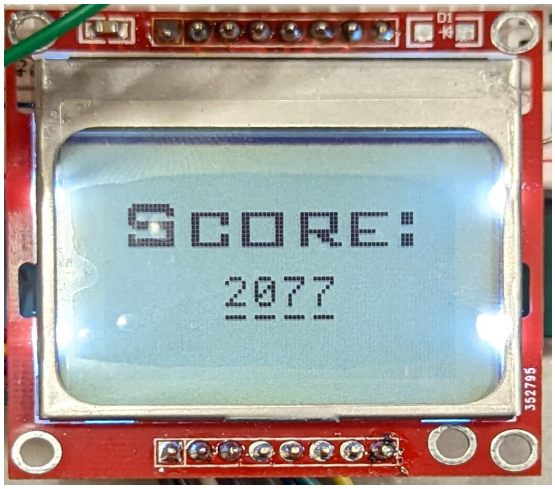


‘G’ - tone Frequency: 392 Hz



LCD Pictures:

Score Screen:



Game Screen: